UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

GUSTAVO NASCIMENTO COSTA

EMAIL MARKETING COM JAVA E ANGULAR UTILIZANDO SERVIÇO AMAZON

FLORIANÓPOLIS

2017

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

EMAIL MARKETING COM JAVA E ANGULAR UTILIZANDO SERVIÇO AMAZON

GUSTAVO NASCIMENTO COSTA

Trabalho de conclusão de curso apresentado
como parte dos requisitos para obtenção do
grau de Bacharel em Sistemas de Informação

FLORIANÓPOLIS

2017

Gustavo Nascimento Costa

EMAIL MARKETING COM JAVA E ANGULAR UTILIZANDO SERVIÇO AMAZON

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Fernando Augusto da Silva Cruz

Banca Examinadora:

_____

Prof. Fernando Augusto da Silva Cruz

_____

Prof. João Bosco Mangueira Sobral

_____

Prof. Leandro José Komosinski

“Dedico este trabalho aos meus pais, motivadores das minhas conquistas e responsáveis por este momento.”

AGRADECIMENTOS

Agradeço aos professores que fizeram parte dessa trajetória, em especial ao meu orientador, Professor Fernando Augusto da Silva Cruz.

SUMÁRIO

LISTA DE FIGURAS

RESUMO

Este trabalho propõe a implementação de uma aplicação de E-mail marketing utilizando Java e uma base de dados MySql, e o serviço simple email services de e-mails da Amazon, na parte do servidor. No cliente é utilizada a framework Angular. O projeto tem como objetivo demonstrar todas as tecnologias envolvidas, utilizadas em conjunto para o desenvolvimento da aplicação.

Palavras-chave: e-mail marketing, Java, Angular, Amazon SES

ABSTRACT

This work describes the implementation of an email marketing application using Java and MySQL database, and the Amazon's simple email services on the server side. Angular framework is used on the client side. The project objective is to show the involved technologies, used together to accomplish the project.

Keywords: email marketing, Java, Angular, Amazon SES

1. Introdução

E-mail marketing é cada vez mais reconhecido como uma ferramenta de marketing eficaz na Internet. Mantendo um canal de comunicação com seus clientes, baixo custo e alta taxa de resposta, muitas empresas têm investido em ferramentas de E-mail marketing. Cada vez mais sofisticadas e inteligentes, proporcionam retorno imediato da aceitação de campanhas de marketing por e-mail.

Um grande obstáculo são os *spams,* e-mails recebidos sem autorização que muitas vezes podem trazer riscos à segurança dos destinatários.

Neste projeto, foi desenvolvida uma aplicação de E-mail Marketing Web com interface HTML5 utilizando-se Angular no cliente, e Java em máquina virtual no servidor, utilizando serviços de e-mail da Amazon.

1.1 Motivação

Desenvolver uma ferramenta de E-mail marketing que vise solucionar uma questão crucial sobre uma campanha de e-mail. Saber com exatidão a porcentagem dos e-mails enviados que tiveram sucesso em fazer o destinatário clicar no link contido e ser direcionado ao objetivo.

Além do interesse e vontade em aprofundar conhecimentos nas tecnologias

utilizadas nesta aplicação web, somado ao ótimo serviço de envio de e-mails provido pela Amazon que respeita todas as normas e validações para E-mail marketing, reduzindo as chances de que os e-mails sejam considerados lixo virtual pelos filtros de *spam* padrões do mercado.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo geral é a realização de um estudo das tecnologias, linguagens e serviços utilizados, com a subsequente aplicação do conhecimento adquirido no desenvolvimento de uma aplicação web de E-mail marketing.

1.2.2 Objetivos Específicos

Implementar uma aplicação web de E-mail marketing.

Implementar um servidor remoto com acesso a uma base de dados MySql hospedada em uma máquina local.

Implementar a comunicação do servidor remoto com a aplicação cliente e com o serviço de envio de e-mail da Amazon.

1.3 Justificativa

Atualmente o E-mail marketing é de extrema importância para as empresas, objetivando manter relação com possíveis novos clientes ou já existentes, informando sobre promoções, eventos, novos produtos e serviços.

O marketing pode ser um fator decisivo entre o sucesso e fracasso de uma empresa. Mesmo com um produto melhor que seu concorrente, uma empresa pode perder mercado caso seu concorrente execute de forma mais efetiva suas campanhas de marketing.

Dentro desse contexto a aplicação desenvolvida poderá ser utilizada de qualquer dispositivo com navegador compatível com HTML5, conectado à internet, sem instalação de *softwares* adicionais, suprindo uma demanda básica de enviar e-mails de uma campanha marketing a uma lista de clientes previamente importada.

1.4 Premissas

A solução a ser desenvolvida neste projeto é de caráter digital: um conjunto de *softwares* e serviços integrados visando um objetivo final. Para termos um cenário

realístico, o usuário terá que se cadastrar no sistema com um e-mail válido e importará uma lista de contatos de e-mails para enviar sua campanha de marketing exemplo. Após todos os e-mails enviados será possível analisar quantos usuários clicaram no link da campanha de marketing, tendo um bom indicador de taxa de sucesso da campanha exemplo.

2. Fundamentação Teórica

2.1 Java EE 8

Acompanhando o crescimento exponencial da tecnologia da informação, há uma crescente necessidade de aplicações distribuídas, transacionais e portáveis que não pequem nos quesitos velocidade, segurança e estabilidade. Com a plataforma Java Enterprise Edition 8 (Java EE 8), o desenvolvimento de aplicações complexas nunca foi tão prático e rápido. O objetivo da mesma, é proporcionar aos desenvolvedores, um conjunto poderoso de APIs para melhor performance, redução do tempo de desenvolvimento e da complexidade da aplicação.

A plataforma Java EE 8 utiliza um modelo de programação simplificado, onde arquivos de

configuração XML, por exemplo, tornam-se opcionais. Em seus lugares, o desenvolvedor pode inserir estas informações em forma de anotações, marcadas pela diretiva @, diretamente no código Java. O Java EE encarrega-se de configurar o componente na fase de deploy e em tempo de execução. Com as anotações, especificamos com facilidade as configurações, antes feitas separadamente em arquivos XML, diretamente ao lado da classe, atributo ou método afetado.

Através da plataforma Java EE, declaramos também injeções de dependências a qualquer componente através de rápidas anotações, evitando a necessidade da implementação de lookups de recursos da aplicação. Injeções de dependências são permitidas em containers EJB, WEB e aplicações cliente, tornando automática a inserção de referências de outros componentes e recursos apenas com o uso de anotações.

2.2 MySQL

O MySQL é um Sistema de Gerenciamento de Banco de Dados, ou simplesmente SGBD. Ele faz uso da linguagem SQL para manipulação dos dados armazenados e configurações.

MySQL é um grande sucesso nos dias de hoje, incluso nos principais pacotes de hospedagem de sites na internet. Parte de seu sucesso está ligado a fácil integração com o PHP, linguagem de programação utilizada na WEB. Além disso, o MySQL é um dos poucos SGBDs gratuitos que competem com os principais SGBDs privados, como Oracle e SQLServer.

O MySQL é um software livre com base na GPL (Licença Pública Geral) de fácil uso, leve, apresentando excelente desempenho e estabilidade e baixa exigência de recursos de hardware. A portabilidade também se destaca, sendo suportado praticamente
por qualquer plataforma atual. APIs para integração com a maioria das linguagens de programação estão disponíveis, como Delphi, C/C++, Python, PHP, ASP, Ruby e o próprio Java. Ferramentas gráficas altamente qualificadas estão disponíveis gratuitamente para a manipulação do SGBD, a exemplo do MySQL Workbench.

Em termos lógicos, o MySQL suporta controle transacional, triggers, cursores, procedures, functions, replicação de dados e contempla a utilização de diversos Motores de Armazenamento (Storage Engines), como o InnoDB, Falcon, CSV, MyISAM, entre outros.

Reconhecido mundialmente pelo seu desempenho e robustez, inúmeras empresas de grande porte optaram por seu uso, a exemplo do Banco Bradesco, HP, NASA, Nokia, Sony, Cisco, Wikipedia, entre outros.

2.3 Angular

Angular é um framework JavaScript open-source, que auxilia na execução de single-page applications. Tem como objetivo facilitar o desenvolvimento de aplicativos que

podem ser acessados via navegador web, foi construído sob o padrão model-view-view-model (MVVM), em um esforço para facilitar tanto o desenvolvimento quanto o teste dos aplicativos.

A biblioteca lê as diretivas contidas nas tags HTML especiais e então executa a diretiva na qual esta tag pertence, e faz a ligação entre a apresentação e seu modelo, representado por variáveis JavaScript comuns. O valor dessas variáveis JavaScript podem ser setadas manualmente, ou via um recurso JSON estático ou dinâmico.

Angular segue o padrão MVC da engenharia de Software e encoraja o baixo acoplamento entre apresentação, dados e componentes lógicos. Usando injeção de dependência, Angular traz serviços comumente designados ao lado servidor da aplicação, como controllers para os componentes visuais, para o lado cliente da aplicação. Consequentemente, o peso do backend é radicalmente reduzido, levando à aplicações mais leves.

2.5 Amazon Simple Email Service – SES

O Amazon Simple Email Service trata-se de um serviço de envio de e-mail que elimina a complexidade de manter um serviço de e-mail próprio. Com baixo custo e sendo disponibilizado um serviço que pode ser usado em larga escala, garantindo que os emails sejam entregues com êxito. Seguindo os rigorosos padrões de Internet Service Provider (ISP) sobre o conteúdo do e-mail.

Sendo possível ter acesso a uma infraestrutura de e-mail de alta qualidade e escalável para envio de e-mail aos seus clientes de modo eficiente e econômico.

3 Ferramentas

3.1 MySQL WorkBench

O MySQL WorkBench é uma ferramenta visual completa para desenvolvedores, arquitetos e DBA`s que fazem uso dos servidores de banco de dados MySQL. Podemos classificá-la em 3 funcionalidades majoritárias: Modelagem de Dados, Manipulação de

Dados e Administração das Bases de Dados. A ferramenta é atualmente distribuída entre os sistemas operacionais mais populares, Windows, Linux e Mac OS, mostrando-se uma excelente opção para os usuários MySQL.

Modelagem de Dados

O MySQL WorkBench permite ao desenvolvedor visualizar, modelar, gerar e administrar base de dados. Ele inclui todos os recursos necessários para criação dos

mais complexos modelos ER, engenharia direta e reversa, criação e manutenção de documentações.

Manipulação de Dados

O MySQL WorkBench traz ferramentas visuais para criação, execução e otimização de queries SQL. Pode-se optar por deixar de lado a escrita de scripts, uma vez que é possível fazer quase tudo através da interface, desde a criação e alteração de tabelas, relacionamentos e rotinas, como o autoincrement, até pequenos inserts.

Administração das Bases de Dados

O MySQL WorkBench proporciona um console visual onde podemos administrar os servidores MySQL com facilidade. Desenvolvedores e DBAs podem ser usados para configuração de servidores, administração de usuários e visualização de dados estatísticos sobre a base.

3.2 Eclipse

Eclipse é um Ambiente Integrado de Desenvolvimento (IDE) de código aberto desenvolvido em Java. Apesar da possibilidade de que podemos utilizar a plataforma eclipse no desenvolvimento de linguagens como C, C++ e PHP através do uso de plugins, seu principal foco é o desenvolvimento com a linguagem Java, sendo a uma das IDEs mais utilizada no mundo para tal propósito.

Uma de suas principais características é a forte orientação ao desenvolvimento

baseado em plug-ins, proporcionando amplo suporte ao desenvolvedor, que encontra, nos mesmos, diferentes soluções para atender suas diferentes necessidades.

3.3 Maven

Maven, um projeto da Apache Software Foundation, é uma ferramenta para gerenciamento e automação de projetos Java. Similar à ferramenta Ant, possui um modelo de configuração mais simples, baseado no formato XML e unificado em um único arquivo, o Project Object Model, ou simplesmente pom.xml. Nele está descrito todo o processo de construção de um projeto, suas dependências, módulos adicionais, componentes e sequência de construção. Dentre as tarefas executáveis disponibilizadas pelo Maven, as mais conhecidas e utilizadas são a compilação e o empacotamento de projetos, tendo no pom.xml, todas as informações necessárias para seu sucesso. Quaisquer dependências referenciadas no arquivo de configuração, por exemplo, serão automaticamente baixadas no momento da compilação.

4. Especificação do Sistema

4.1 Diagrama de Casos de Uso

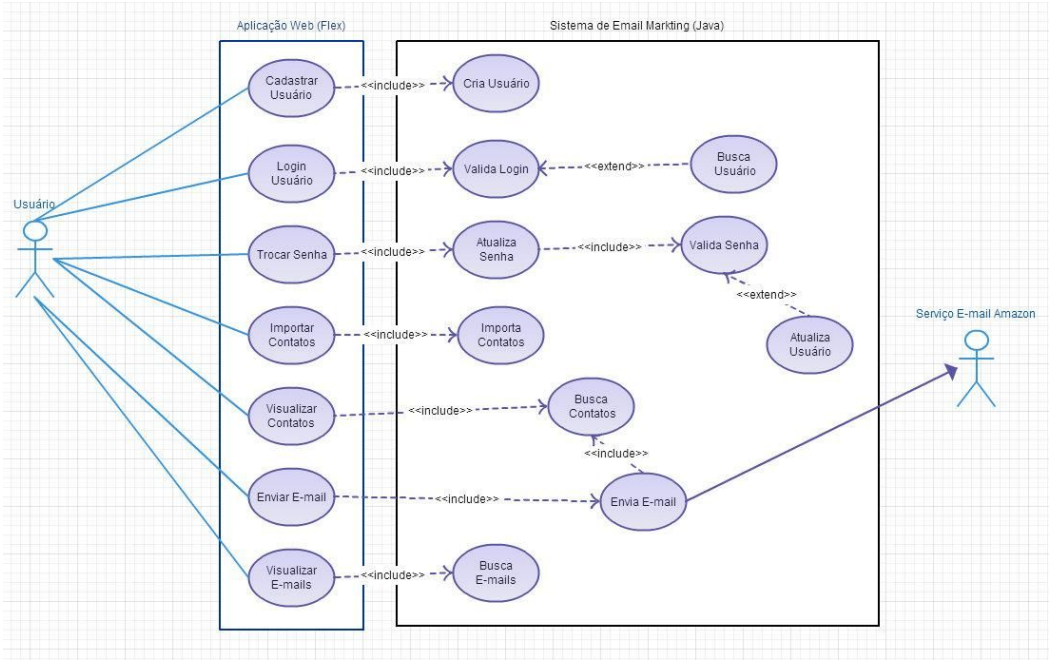

Figura 4.1 - Diagrama de casos de uso

Notamos a necessidade de dois softwares diferentes e integrados para a implementação

do sistema. Sendo a aplicação Web (Angular) responsável por receber as ações do usuário, realizar validação básica da requisição e enviar a chamada ao Sistema de E-mail Marketing (Java) remoto.

Quando a ação desejada for o envio de e-mail o Sistema de E-mail Marketing enviará o comando ao Serviço de E-mail da Amazon que se encarregará de efetivamente

enviar o e-mail aos destinatários.

4.2 Requisitos do Sistema

Visão Geral
O presente documento de requisitos, descreve um sistema web envio de e-mails para sua lista de contatos. Tendo uma analise de sucesso de cada e-mail enviado a partir do número de destinatários que clicarem no link contido no e-mail.

Sistema de E-mail Marketing
O sistema deve disponibilizar serviços web e interfaces remotas para ativação das funcionalidades da solução proposta.

Sistema de E-mail Marketing			
ID	**Descrição**	**P**	**D**
RF01	O Sistema deve permitir que qualquer pessoa que possuir um e-mail válido possa se cadastrar.	Alta	
RF02	O Sistema deve possibilitar o usuário importar sua lista de contatos em formato compatível com o Gmail.	Alta	RF01
RF03	O Sistema deve possibilitar o envio de e-mail para sua lista de contatos.	Alta	RF02
RF04	O Sistema deve manter o histórico de e-mails enviados	Média	RF03
RF05	O Sistema deve armazenar a quantidade de destinatários que interagiram com o e-mail. Exibindo a taxa de sucesso.	Média	RF03
RF06	O Sistema deve disponibilizar serviço de autenticação de login e senha de um usuário. Retornando todos o dados necessários do respectivo usuário.		RF01
RF06	O Sistema de disponibilizar a funcionalidade de troca de senha ao usuário cadastrado.		RF01
Requisitos Não Funcionais			
RNF01	O sistema deve contar com o SGBD MySQL para gerenciamento e armazenamento dos dados.	Alta	--
RNF02	O sistema deve ser desenvolvido na linguagem de programação Java.	Alta	--

Aplicação Web (Angular)
Aplicação cliente, efetua chamadas remotas ao sistema de E-mail Marketing para utilização de suas funcionalidades e retorno de dados.

Aplicação Web (Angular)			
ID	**Descrição**	**P**	**D**
RF07	A aplicação deve conter uma tela de cadastro / login e enviar uma chamada remota ao sistema de E-mail Marketing para autenticação contendo os dados inseridos no formulário.	Alta	
RF08	Após o login, a aplicação deve realizar uma chamada remota ao sistema de E-mail Marketing retornando as informações do usuário e exibindo a tela principal da aplicação.	Alta	RF07
RF09	A aplicação deve conter uma tela para alteração de senha do usuário logado.	Alta	RF07
RF10	A aplicação deve conter uma tela para visualização do histórico de e-mails enviados.	Média	RF12
RF11	A aplicação deve conter uma tela para importação e visualização dos contatos importados.	Média	RF12
RF12	A aplicação deve conter uma tela para envio de e-mail para sua lista de contatos.	Alta	RF07
Requisitos Não Funcionais			
RNF03	O aplicativo deve ser desenvolvido com as linguagens HTML5 e TypeScript utilizando Angular.	Alta	--

Legenda:
P - Prioridade
D - Depende de

4.3 Modelo Entidade Relacionamento dos Dados

Com as metas para o objetivo final bem especificadas pelo documento de requisitos, podemos começar a implementação da solução do projeto. Seguindo a ordem de dependências proposta pelo documento de requisitos, iniciaremos o desenvolvimento pelo Sistema de E-mail Marketing. Para tal, faz-se necessário a modelagem e implementação da base de dados MySQL encarregada do controle e armazenamento dos registros do sistema.

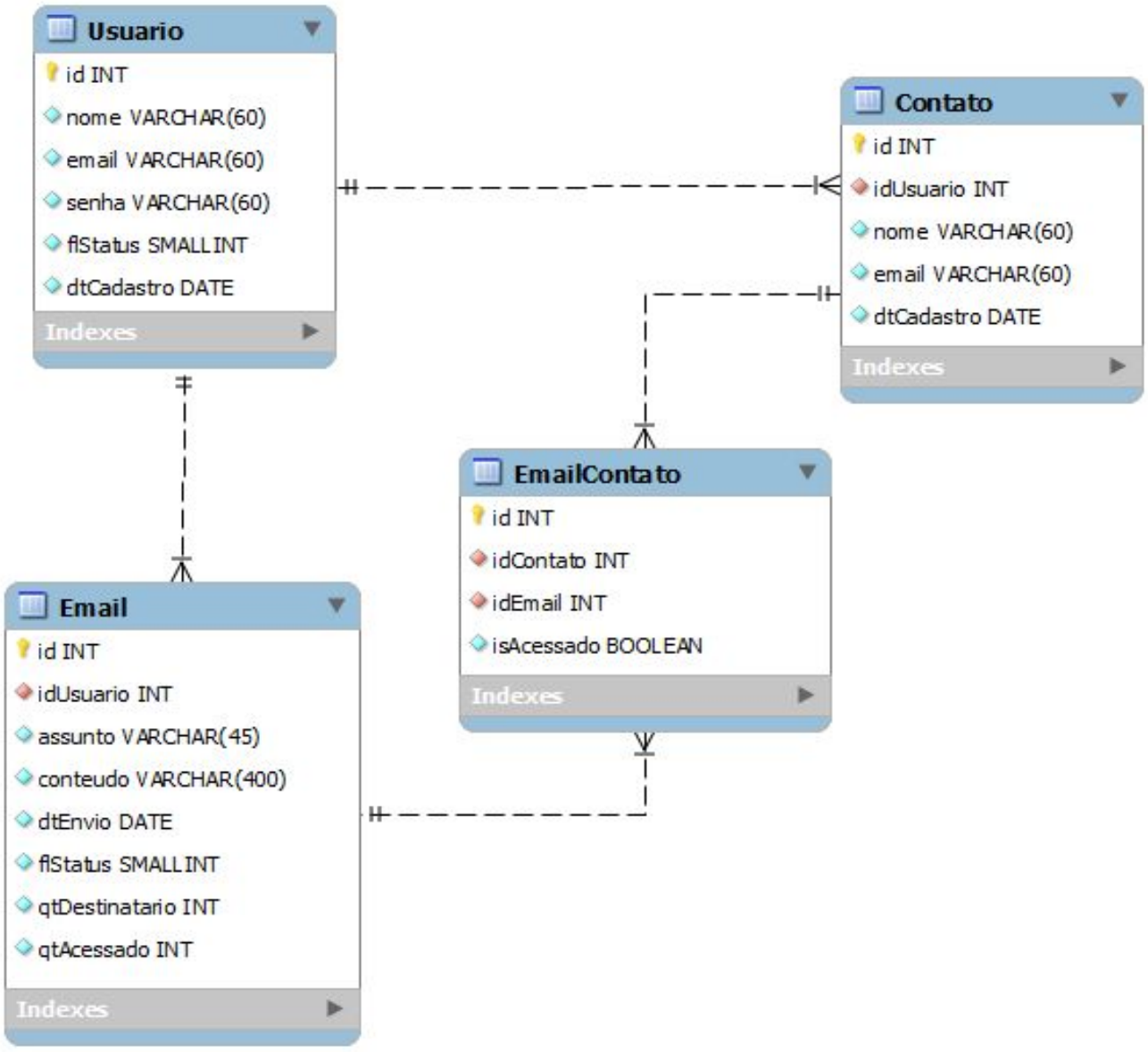

Figura 4.3 - Modelo entidade relacionamento.

A base de dados, acima modelada, abrange as funcionalidades chave para que se atenda de forma integral, todos os requisitos do sistema. Com o modelo de dados e o documento de requisitos em mãos, obtidos durante o planejamento do projeto, temos agora, os fundamentos necessários para a próxima etapa do projeto, a implementação.

4.4 Wireframes

Com as metas para o objetivo final especificadas pelo documento de requisitos, podemos começar a implementar a solução do projeto. Seguindo a ordem de dependências proposta pelo documento de requisitos, iniciaremos o desenvolvimento pelo Sistema de E-mail Marketing. Para tal, faz-se necessário a prototipação das telas do sistema.

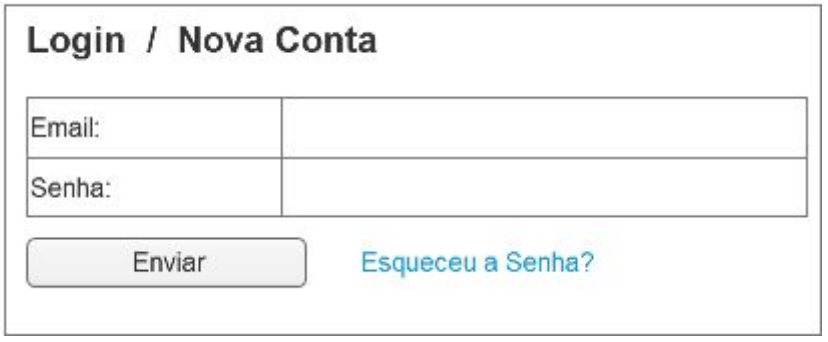

Figura 4.4.1 -Tela de Login

A tela a cima representa o estado inicial de acesso ao sistema. Tendo a opção de login ou criação de uma nova conta a partir de um E-mail e Senha.

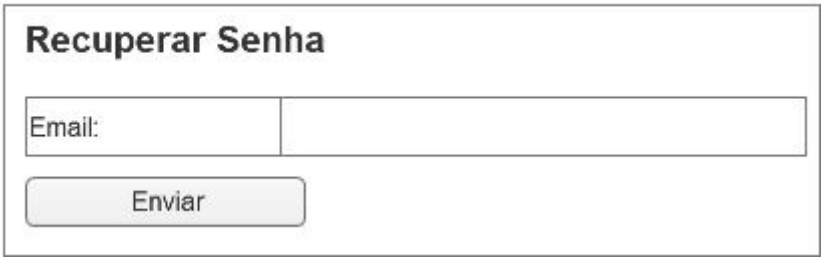

Figura 4.4.2 - Tela de Recuperar Senha

Agora temos a tela responsável por recuperar a senha do Usuário. Nela será

preenchido o E-mail onde o usuário receberá uma nova senha.

Figura 4.4.3 - Tela de Emails Enviados

A tela a cima representa a listagem de e-mail com campanhas de marketing enviadas, sendo possível analisar para quantos contatos foram enviados o e-mail e a taxa de sucesso. A taxa de sucesso é medida pelo número de contatos que clicaram no link disponibilizado no e-mail.

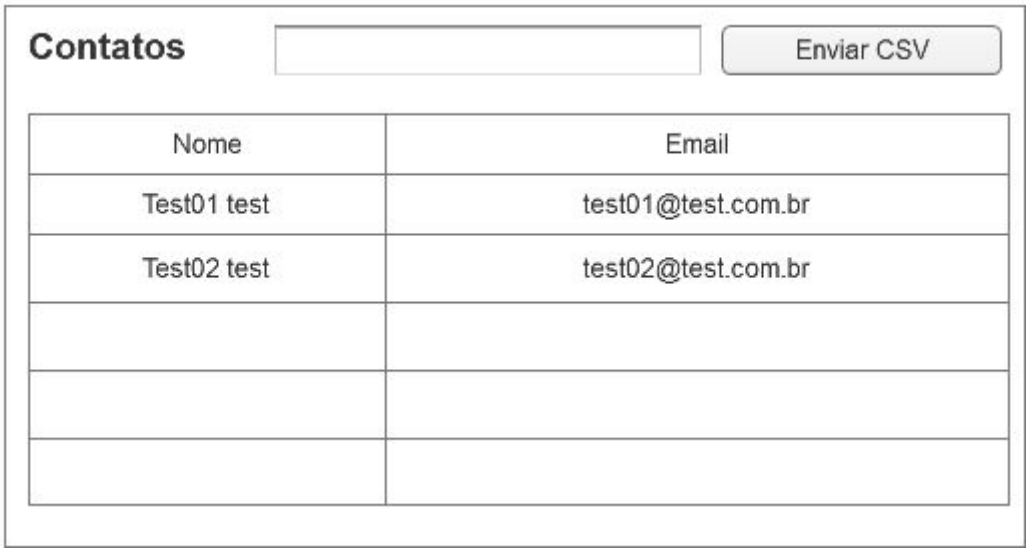

Figura 4.4.4 - Tela de Contatos

Aqui temos representada a tela que possibilitará o usuário importar um arquivo CSV com contatos que receberão suas campanhas de e-mail de marketing.

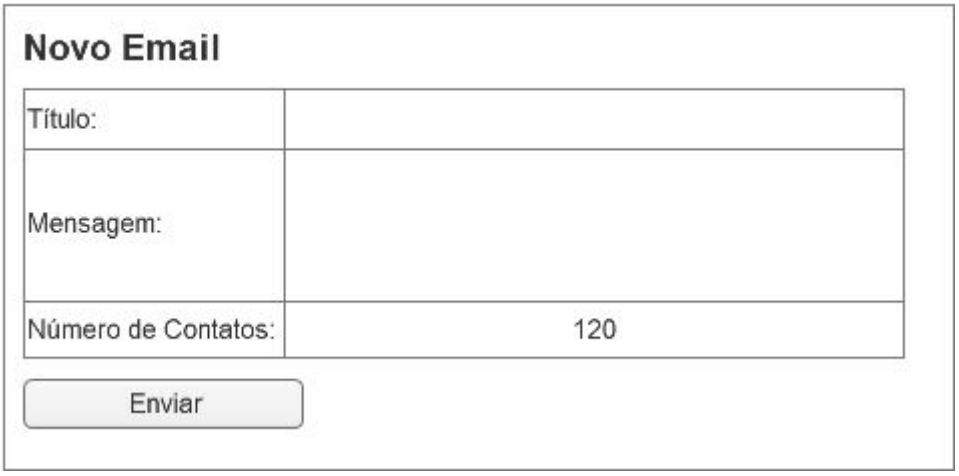

Figura 4.4.5 - Tela de Novo Email

Acima a tela de envio de e-mail. Sendo necessário o preenchimento do título e mensagem. A tela informa o número de contatos que receberão o e-mail de marketing.

5. Implementação

Observando a figura do Diagrama de casos de Uso, identificamos com clareza as três diferentes partes integrantes do sistema. São elas: Email Marketing Api, Frontend e Sistema do Banco de dados. A ordem de implementação adotada segue, respectivamente, a esta descrição.

5.1 Email Marketing Api

O Email Marketing Api tem o intuito de provir todas as funcionalidades necessárias. Sendo que o mesmo contém um servidor dedicado para rodar a aplicação em questão. O sistema será desenvolvido na linguagem Java de programação e compilado em um pacote WAR, para posterior deploy em servidor dedicado. As funcionalidades abrangidas pela aplicação dizem respeito apenas às questões críticas para o alcance do objetivo final deste projeto. Trabalhando com a arquitetura MVC (Model, View, Controller), isolando a interface do usuário (Frontend) da lógica do sistema.

5.1.1 Configuração do Ambiente

Para o desenvolvimento do sistema, utilizaremos a ferramenta Eclipse com a adição de dois plugins fundamentais: JBoss Tools e Maven Integration for Eclipse. A primeira nos serve para integrar o servidor de aplicação jBoss 7.1 ao Eclipse, permitindo o deploy do sistema na própria ferramenta, facilitando assim a depuração da aplicação. O segundo integra a ferramenta Maven à IDE, fazendo com que novas dependências apontadas no projeto sejam baixadas automaticamente. Chamaremos nosso novo projeto de EmailMarketingApp. Este será implementado com a plataforma Java EE 8, fazendo uso de Java Beans e injeções de dependência de recursos.

5.1.2 Base de Dados MySQL

Cabe agora, a implementação da base de dados MySQL, seguindo os moldes projetados na figura 5.3 - Modelo Entidade Relacionamento de Dados. Chamaremos a base de dados de ‘main’, o schema da aplicação de ‘emailmarketing’ e criaremos dois usuários para o acesso - o usuário ‘gustavo’ com todas permissões, para gerenciamento da base e o usuário ‘application’ com permissões de consultas e inserção de dados, para o acesso a ser configurado na aplicação.

5.1.3 Camada de Domínio

Como primeiro passo da implementação de nossa aplicação, faremos o mapeamento das classes Java relativas às tabelas da base de dados.

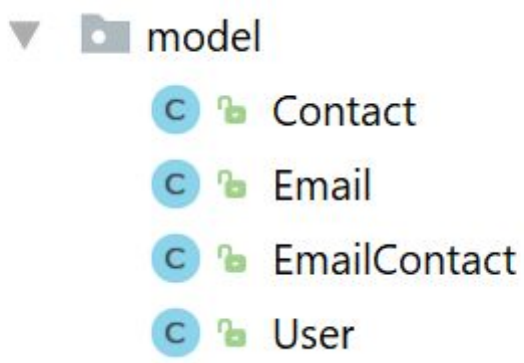

Figura 5.1.3.1 - Camada de Domínio

No pacote ‘ufsc.emailmarketing.model’ foram criadas as classes de mapeamento da base. Identificamos as tabelas, nela existentes, em: ‘Contact.java’, ‘Email.java’, ‘EmailContact.java’ e ‘User.java.’ Percebemos, no entanto, que elas não estão a sós. No pacote ‘ufsc.emailmarketing.support’ , ainda existe outra classe - ‘AbstractEntity.java’. Como o próprio nome sugere, é uma classe abstrata estendida por todas entidades de mapeamento, com o intuito de facilitar posterior abstração nas regras da camada de negócio.

Para o mapeamento das tabelas como classes e de suas colunas como atributos, fazemos uso de anotações da API javax.persistence, pertencente a plataforma Java EE.

```
@Entity
@Table(name = "email")
public class Email extends AbstractEntity {

    private String assunto;

    private String conteudo;

    private Date dtEnvio;

    private Integer flStatus;

    private Integer qtDestinatario;

    private Integer qtAcessado;

    @ManyToOne
    @JoinColumn(name = "idUsuario")
    private User usuario;
```

Figura 5.1.3.2 - Camada de Domínio - Email

As anotações @Entity e @Table, classificam a classe Email como uma entidade relacionada a tabela ‘email’ da base de dados. As anotações @Column, relacionam os atributos da classe às colunas da tabela na base de dados. Ainda para as colunas, temos as anotações @Id, apontando o atributo como sendo o identificador da tabela, e @GeneratedValue, para que seja especificada a maneira em que valores numéricos automáticos serão gerados, dada a existência de um auto increment na coluna da tabela.

5.1.4 Camada de Controle

Com as classes de domínio já implementadas, partimos para o desenvolvimento das classes controladoras, contendo as regras de negócio da aplicação.

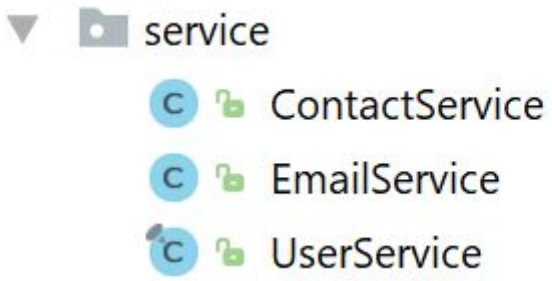

Figura 5.1.4.1 - Camada de Controle

Observando a figura acima, podemos perceber que cada classe de controle é um Java Bean. Nota-se também, como nas classes de domínio, a implementação de uma classe de controle abstrata, está estendida pelas demais classes de controle. A disponibilização dos métodos em interfaces remotas que veremos mais à frente, nos permitirá, entre outras coisas, a integração do Frontend (interface que interage com o usuário/cliente), fechando o ciclo da arquitetura MVC.

```
public abstract class AbstractService<E extends BaseEntity, R extends JpaRepository<E, Long>> implements BaseService<E> {

    @Autowired
    protected R repo;

    protected static Logger log = LoggerFactory.getLogger(AbstractService.class);

    @Override
    public E save(E entity) { return repo.save(entity); }

    @Override
    public Iterable<E> save(Iterable<E> entity) { return repo.save(entity); }

    @Override
    public Boolean delete(Iterable<E> entity) {
        repo.delete(entity);
        return Boolean.TRUE;
    }

    @Override
    public Page<E> list(Pageable pageable) { return repo.findAll(pageable); }

    @Override
    public Boolean delete(Long id) {
```

Figura 5.1.4.2 - Camada de Controle Classe Abstract

A classe da figura acima abstrai os métodos de atualização de registros, save(), exclusão de registros, delete() e de listagem de registros, list() comuns a todos os beans de controle. Tais métodos são declarados nas interfaces abstratas implementadas nesta classe. Para que a abstração seja possível, fazemos uso da classe de domínio abstrata descrita no tópico anterior (AbstractEntity), declarando R como um parâmetro de classe, sendo que R é uma classe a qual estende o AbstractEntity que implementa BaseEntity.

Temos ainda na classe de controle abstrata a declaração do EntityManager (repo), atributo que será herdada pelas demais classes controladoras.

```java
@Service
public class EmailService extends AbstractService<Email, EmailRepository> {

    @Autowired
    private ContactRepository contactRepository;

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private EmailContactRepository emailContactRepository;

    public Page<Email> listEmails(Long userId, Pageable pageable) throws Exception {
        return this.repo.findByUserIdAndDeleted(userId, deleted: false, pageable);
    }

    public void sendEmail(Long userId, Email email) throws Exception {
        User user = userRepository.findOne(userId);

        List<Contact> contatcs = this.contactRepository.findByUserIdAndDeleted(userId, deleted: false);
        LinkedList<String> recipients = new LinkedList<~>();

        email.setUsuario(user);
        final Email emailToSend = this.repo.save(email);
```

Figura 5.1.4.3 - Camada de Controle EmailService

Na figura acima temos parte do código fonte da classe controladora EmailService. Esta, assim com as demais, estende a classe controladora abstrata, passando como parâmetro a classe de domínio Email. Acima da declaração da classe, temos a anotação @Service, a qual configura a mesma como um Java Bean. Em seus métodos temos a interação com a base de dados e a lógica das regras de negócio, quando necessário.

5.1.5 Serviços Remotos

Com as classes de controle devidamente implementadas, a informação precisa chegar ao servidor para poder ser processada na lógica da aplicação. Este é o próximo passo na implementação do sistema. Disponibilizar serviços web que permitam a interação entre cliente (Frontend) e servidor.

Voltando nossa atenção novamente a figura 4.3 - Diagrama de casos de uso, percebemos, com clareza, os serviços que devem ser disponibilizados para que o cliente possa existir e cumprir com sucesso o seu objetivo:

- **Autenticação Remota** - Como a aplicação cliente inicia sua interação com o usuário? Com uma habitual tela de login, onde serão coletados o e-mail e a senha personalizada do usuário. Devemos implementar e disponibilizar um serviço, que receba essas duas informações e faça a autenticação remota das mesmas, retornando o sucesso ou insucesso da operação.
- **Importação e listagem de contatos** - A partir a tela de contatos será possível importar e listar os contatos importados. Devemos implementar e disponibilizar um serviço que execute a tarefas citadas.
- **Envio e listagem de emails** - A partir da tela de emails será possível enviar um novo email para todos os seus contatos importados anteriormente e listar os emails já enviados. Devemos implementar e disponibilizar um serviço que execute a tarefas citadas.

- **Marcar email como sucesso** - A partir do link no conteúdo do email será marcado o email como sucesso. Teve o objetivo de clique no link alcançado.

Para disponibilização dos serviços, precisaremos de interfaces remotas RESTs. Esses chamados de controllers.

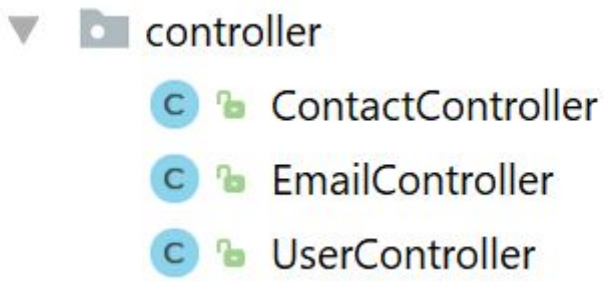

Figura 5.1.5.1 - Camada de Serviços Remotos

Observando a figura acima, podemos perceber que cada classe é um interface remota REST. Nota-se também, como nas classes de domínio, a implementação de uma classe abstrata, estendida pelas demais classes.

```java
public abstract class AbstractController<E extends BaseEntity,
                                         S extends BaseService<E>> {

    protected static Logger log = LoggerFactory.getLogger(AbstractController.class);

    private static final SuccessResponse SUCCESS_RESPONSE = new SuccessResponse();

    @Autowired
    protected S service;

    @RequestMapping(method = RequestMethod.GET)
    public Page<E> all(@PageableDefault(page = 0, size = 10) Pageable pageable) { return service.list(pageable); }

    @RequestMapping(method = RequestMethod.POST)
    public E create(@RequestBody E e) throws IOException {
        return service.save(e);
    }

    @RequestMapping(method= RequestMethod.GET, value = "/{id}")
    public Object read(@PathVariable String id) {
        final Long entityId = Long.valueOf(id);
        final E entity = service.get(entityId);
        return entity == null ? new ErrorMessage(entityId, message: "no matches found") : entity;
    }
```

Figura 5.1.5.2 - Camada de Serviços Remotos Classe Abastract

A classe da figura acima abstrai os métodos de CRUD de registros, create(), retrieve(), update, delete() e listagem de registros, list() comuns a todos os beans de remotos. Tais métodos são declarados nas interfaces abstratas implementadas nesta classe. Para que a abstração seja possível, fazemos uso da classe de domínio abstrata descrita no tópico anterior (AbstractEntity), declarando E como um parâmetro de classe, sendo que E é uma classe a qual estende o AbstractEntity que implementa BaseEntity. Também fazemos uso da classe de abstrata de controle AbastractService que implementa BaseService declarando como S.

Temos ainda na classe REST remota abstrata a declaração da controladora (service), atributo que será herdada pelas demais classes.

```
@RestController
@RequestMapping("/emails")
public class EmailController extends AbstractController<Email, EmailService> {


    @RequestMapping(value = "/send", method = RequestMethod.POST)
    public void sendEmail(@RequestBody Email email) throws Exception {
        User user = SecurityUtils.getCurrentUser();
        this.service.sendEmail(user.getId(), email);
    }

    @RequestMapping(value = "/click/{emailContactId}", method = RequestMethod.GET)
    public void markAsClicked(@PathVariable Long emailContactId) throws Exception {
        this.service.markEmailAsClicked(emailContactId);
    }
```

Figura 5.1.5.2 - Camada de Serviços Remotos EmailController

Na figura acima temos parte do código fonte da classe controladora EmailController. Esta, assim com as demais, estende a classe REST remota abstrata, passando como parâmetro a classe de domínio Email e controladora EmailService. Acima da declaração da classe, temos a anotação @RestController a qual configura, a mesma, como um Java Bean Remoto REST e @RequestMapping configurando a url de acesso a esse recurso. Em seus métodos temos a interação a controladora que possui as regras de negócio.

5.1.6 Empacotamento e Deploy

Com o software implementado, é hora de rodarmos a aplicação no servidor de aplicação jBoss 7.1, localizado em nosso servidor físico, previamente configurado. Para tal, é necessário o empacotamento do projeto, a transferência do pacote obtido para o diretório de deploy do jBoss e a inicialização de seu processo.

Para a compilação e o empacotamento do projeto, estaremos utilizando a ferramenta Maven, cujo intuito é realizar uma compilação dinâmica, baixando de forma automática, qualquer biblioteca que tenha sido referenciada em nosso projeto.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>ufsc.emailmarketing</groupId>
    <artifactId>emailmarketing-api</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>

    <name>EmailMarketing-api</name>
    <description>EmailMarketing API</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.3.3.RELEASE</version>
        <relativePath/>
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>1.8</java.version>
        <junit.version>4.11</junit.version>
```

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-rest</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.data</groupId>
        <artifactId>spring-data-rest-hal-browser</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-mail</artifactId>
```

Figura 5.1.6.1 - Maven pom

A figura acima, pom.xml, representa o arquivo de configuração das dependências do maven em nosso projeto. No momento da compilação, a ferramenta buscará por este arquivo na raiz do projeto e se baseará nas informações, nele obtidas, para a compilação. Podemos perceber algumas tags chave no xml, tais como: war, descrevendo o formato do pacote, e , onde estão inclusas todas as dependências utilizadas no projeto.

```
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building EmailMarketing-api 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ emailmarketing-api ---
```

Figura 5.1.6.2 - Compilação Maven

Notamos na imagem, o arquivo pom.xml incluso na raiz do projeto. Executando o comando do maven - mvn clean install - damos início a compilação do projeto, a qual nos resultará no pacote WAR. Com o pacote em mãos, podemos copiá-lo ao diretório de deploy do servidor de aplicação jBoss e subir o processo.

5.2 Email Marketing Frontend

O Email Marketing Frontend tem o intuito de provir todas as funcionalidades necessárias. O sistema será desenvolvido na linguagem HTML e Type Script baseando-se no framework Angular 4. As funcionalidades abrangidas pela aplicação dizem respeito apenas às questões críticas para o alcance do objetivo final deste projeto. O Frontend também trabalha com a arquitetura MVC (Model, View, Controller).

5.2.1 Configuração do Ambiente

Para o desenvolvimento do sistema, utilizaremos a ferramenta Eclipse com a adição de um plugin: AngularJS Eclipse. Integrando o framework Angular à IDE, fazendo com que novas dependências apontadas no projeto sejam baixadas automaticamente. Chamaremos nosso novo projeto de EmailMarketingUI.

5.2.2 Interface

Implementamos uma interface visual para todo o sistema de E-mail Marketing seguindo o que foi projetado nos Wireframes. Nos Wireframes foi descrito todas as funcionalidades de cada tela.

O E-mail Marketing Frontend faz requisições para buscar informações a serem exibidas ou para cadastrar novas informações ao E-mail Marketing Api.

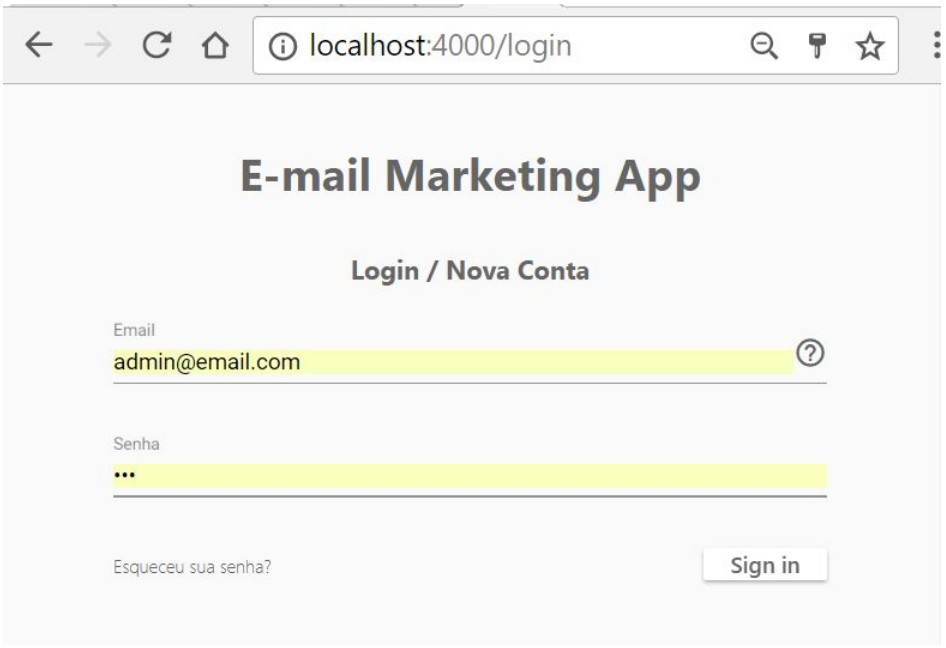

Figura 5.2.2.1 - Tela de Login

Primeiramente o usuário precisará realizar o login ou criar uma conta.

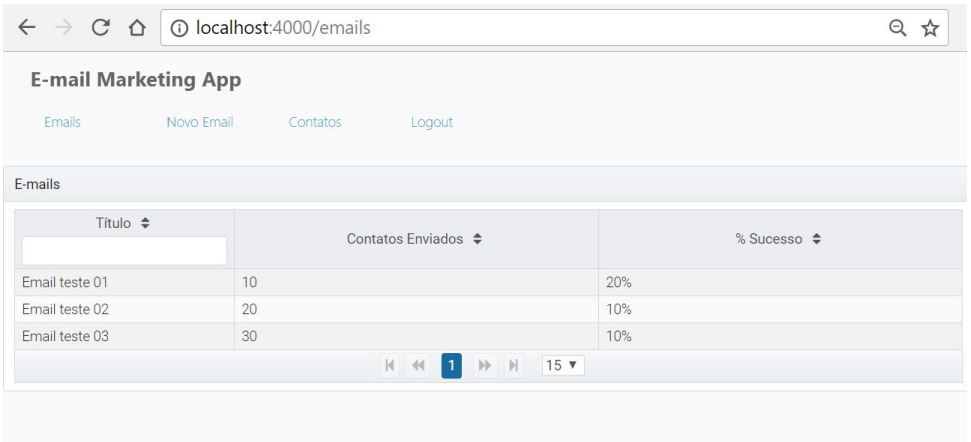

Figura 5.2.2.2 - Tela de Emails

Após feito o login com sucesso. O usuário será redirecionado para a tela Emails. Onde teremos a listagem dos e-mails enviados. Com detalhes sobre quantos contatos o e-mail foi enviado e a porcentagem de sucesso, calculada a partir do número de contatos que clicaram no link recebido no e-mail.

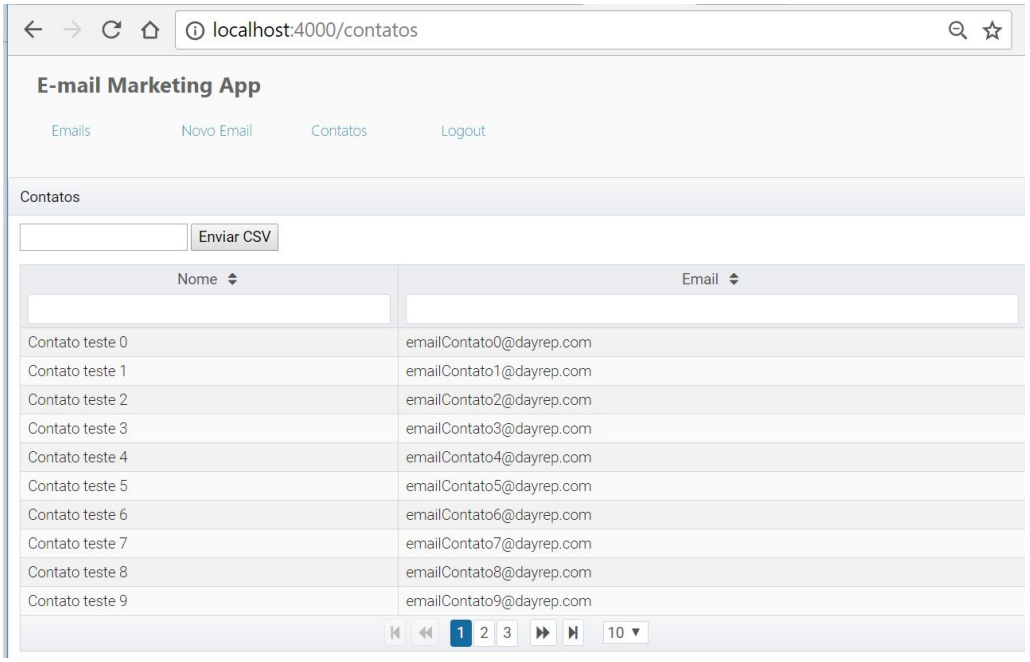

Figura 5.2.2.3 - Tela de Contatos

Será necessário importar um CSV com seus contatos na tela de Contatos.

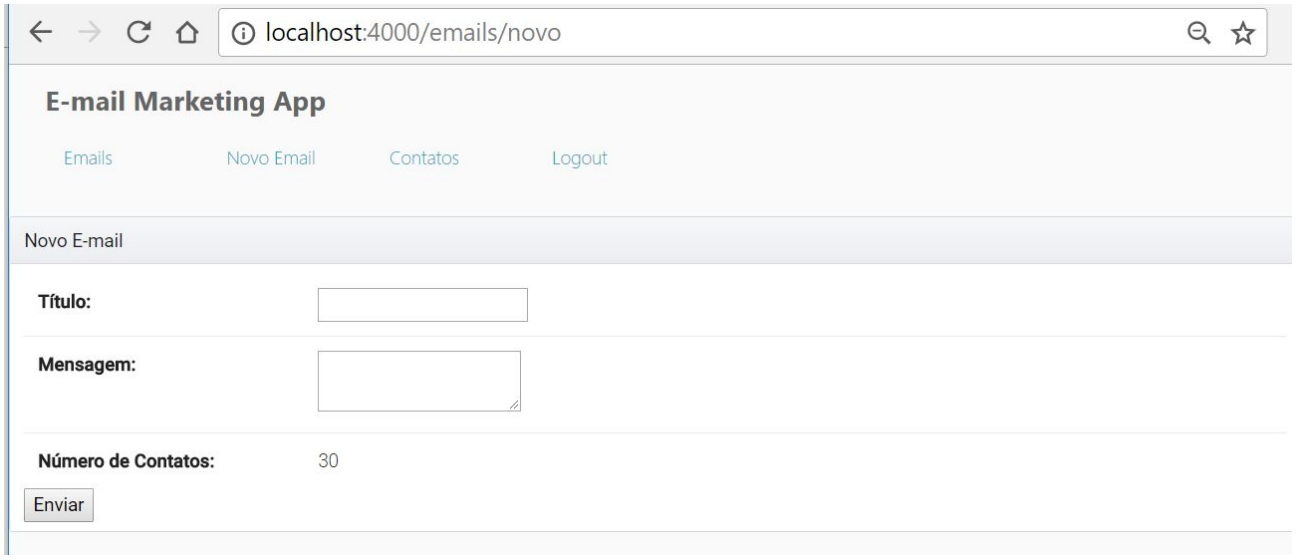

Figura 5.2.2.4 - Tela de Novo Email

Por último o usuário será capaz de enviar um e-mail a todos os contatos importados.

```
<div class="ng-scope">
  <p-panel [header]="'E-mails'" class="col-xs-6 bottom15 width100p right20">
    <div class="panel-body table-responsive relative"
        >
      <p-dataTable [value]="dtoList" class="table table-hover table-striped" #dt sortMode="single"
                   [rows]="15" [paginator]="true" [pageLinks]="5"
                   [rowsPerPageOptions]="[15,30,60,90]">
        <p-column field="titulo" header="Titulo" [sortable]="true"
                  [filter]="true" filterMatchMode="contains"
                  [style]="{'width':'120px'}"></p-column>
        <p-column field="qt_contatos" header="Contatos Enviados" [sortable]="true"
                  [filter]="false" filterMatchMode="contains"
                  [style]="{'width':'200px'}"></p-column>
        <p-column field="porcentagem_sucesso" header="% Sucesso" [sortable]="true"
                  [filter]="false" filterMatchMode="contains"
                  [style]="{'width':'200px'}"></p-column>
      </p-dataTable>
    </div>
  </p-panel>

  <loading-indicator [isLoading]="loading"></loading-indicator>
</div>
```

Figura 5.2.2.5 - HTML tela de Emails

O código html acima é referente a tela de e-mails, emaillist.component.html. Sendo possível notar tags como ‘p-panel’, ‘p-dataTable’ e ‘p-column’ que são referentes a componentes do framework prime-ng. O prime-ng é um framework relativamente completo se tratando de componentes básicos de tela para Angular.

5.2.3 Domínio

É necessário que repliquemos as classes de domínio do sistema E-mail Marketing Api em nossa aplicação Angular, para facilmente receber os dados de resposta obtidos nas requisições remotas.

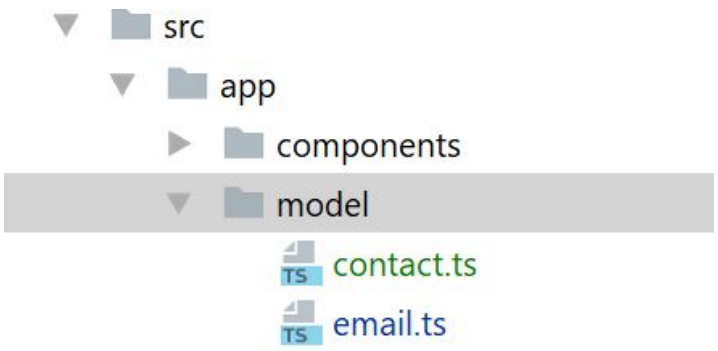

Figura 5.2.3.1 - Camada de Dominio frontend

Notamos através das classes replicadas que, nem todas que lá constam, aqui estão. Não é necessário que implementamos todas as classes de domínio do servidor, mas sim as que são críticas para as funcionalidades da aplicação.

```
export class Contact {
  nome: string;
  email: string;
}
```

Figura 5.2.3.2 - Camada de Dominio frontend Contato

A imagem acima se refere a classe de domínio Contact e seus atributos.

5.2.4 Controle e Requisições Remotas

As classes de controle são criadas para o manuseamento dos componentes da interface e lógica da aplicação.

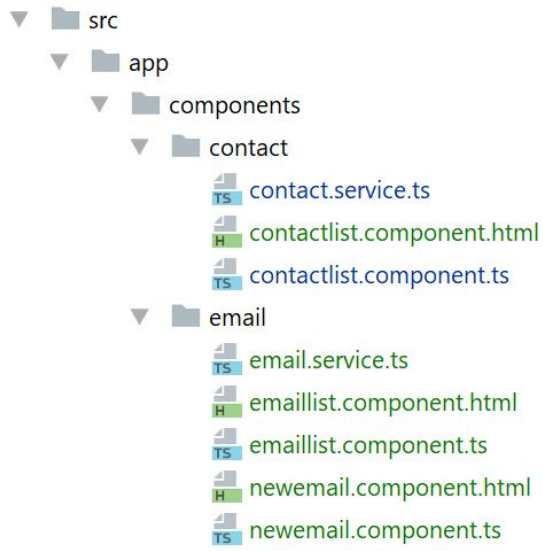

Figura 5.2.4.1 - Camada de Controle e Requisiçoes Remotas

Na imagem acima podemos notar as classes de controle: contactlist.component.ts, emaillist.component.ts e newemail.component.ts. Sendo as classes de Requisições Remotas: contact.service.ts e email.service.ts.

A aplicação em desenvolvimento, está fortemente baseada em requisições remotas. Todos os dados obtidos para apresentação e validação são feitas através das mesmas. Com a listagem de emails não é diferente. Após carregamento da tela uma requisição é enviada ao servidor para buscar os emails e um retorno é obtido com o sucesso ou insucesso da operação.

```
@Component({
  selector: 'app-emaillist',
  templateUrl: './emaillist.component.html'
})
export class EmailListComponent extends LoadingPage implements OnInit {

  dtoList: Email[] = [];

  constructor(private emailService: EmailService) {
    super(false)
  }

  ngOnInit() {
    this.getAll();
  }

  getAll() {
    this.standby();

    this.emailService.getAll().subscribe(
      data => {
        this.dtoList = data;

        this.ready();
      },
```

Figura 5.2.4.2 - Camada de Controle EmailListComponent

Acima temos a classe de controle EmailListComponent onde ao finalizar o

carregamento inicial é feita uma chamada ao método getAll(). Esse buscará no servidor os emails do usuário logado e o retorno obtido será armazenado em dtoList, que será exibido no grid de emails. Tudo a partir da classe de requisição remota EmailService que foi injetada no construtor.

```
@Injectable()
export class EmailService {

  private customerUrl = environment.apiRoot + '/emails';

  constructor(
    private httpOLD: HttpService,
  ) { }


  getAll(): Observable<any> {
    return this.httpOLD.get(this.customerUrl)
      .map((res: Response) => res.json())
      .catch((error: any) => Observable.throw(error.json().error || 'Server error'));
  }

  sendEmail(title: Number, message: Number) {
    var email = { assunto: title, conteudo: message};
    return this.httpOLD.post(this.customerUrl + "/send", email)
      .map((res: Response) => res.json())
      .catch((error: any) => Observable.throw(error.json().error || 'Server error'));
  }
```

Figura 5.2.4.2 - Camada de Requisição Remota EmailService

Acima temos a classe de requisição remota EmailService. Essa responsável por realizar a chamada remota e retornar o resultado com sucesso ou erro. Podemos notar dois métodos remotos: getAll e sendEmail.

6. Conclusão e Trabalhos Futuros

6.1 Conclusão

Ao fim do desenvolvimento, o sentimento que fica é de dever cumprido. O maior objetivo da implementação era ter uma forma de mensurar o sucesso de uma campanha de E-mail marketing. Além de demonstrar o conjunto de tecnologias utilizadas para o desenvolvimento dessa aplicação. Ambos foram atingidos com sucesso, uma vez que as tecnologias foram abordadas e detalhadas, tendo um projeto, rodando perfeitamente todas as suas funcionalidades, como resultado. O projeto pode ser utilizado no mercado? Precisaríamos de uma avaliação de mercado mais profunda. Além de aprimoramentos funcionais

6.2 Trabalhos Futuros

Muito há de ser feito para que tal projeto possa tornar-se realidade. Começando

pela avaliação de mercado citada, para descobrirmos até que ponto sua aplicação é viável. Sendo o retorno positivo, é fácil pensar o que mais deve ser trabalhado. O que fizemos aqui foi uma base, uma aplicação de E-mail Marketing funcional, a partir daí só se faz melhorar: Aperfeiçoar a estrutura, design e usabilidade geral da aplicação; Aumentar o número de funcionalidades da aplicação: Utilizar mais métricas para avaliação do resultados do e-mail, Novas formas de cadastrar contatos, Gerenciamento de listas de contatos. Muito pode ser melhorado e repensado para atender demandas de marketing comercial.

6.3 Fechamento

Para concluir, agradeço a leitura dos que chegaram até aqui. Espero que lhes tenha igualmente agradado com a proposta e descrição da solução desenvolvida no trabalho.

7. Referências Bibliográficas

1. ALEX. Linguagem Java. Urandi - BA, 2008. Disponível em:<http://codigofonte.uol.com.br/artigo/java/artigo-sobre-a-linguagem-java>. Acesso em: 22 mai. 2012.

2. AngularJS Introduction. Disponível em:<https://www.w3schools.com/angular/angular_intro.asp>. Acesso em: 10 Ago. 2017.

3. Angular Documentation. Disponível em:<http://devdocs.io/angular/>. Acesso em: 10 Ago 2017

4. ForumWeb. O que é o MySQL? 2010. Disponível em:<http://www.forumweb.com.br/artigo/79/mysql/o-que-e-o-mysql>. Acesso em: 23 mai. 2014.

5. MySQL. Top Reasons to Use MySQL. Disponível em:<http://www.mysql.com/why-mysql/topreasons.html>. Acesso em: 23 mai. 2014.

6. MySQL. MySQL Workbench 5.2: Visual Database Design. Disponível em:<http://www.mysql.com/products/workbench/design>. Acesso em: 23 mai. 2014.

7. MySQL. MySQL Workbench 5.2: SQL Development. Disponível em:<http://www.mysql.com/products/workbench/dev>. Acesso em: 23 mai. 2014.

8. MySQL. MySQL Workbench 5.2: Administration. Disponível em:<http://www.mysql.com/products/workbench/admin>. Acesso em: 23 mai. 2014.

9. Oracle. The Java EE 6 Tutorial. Disponível em:<http://docs.oracle.com/javaee/6/tutorial/doc/bnaaw.html>. Acesso em: 15 out. 2014.

10. SOLDANO Alessio. JBossWS - CXF WS-ReliableMessaging tutorial. Disponível em:<https://community.jboss.org/wiki/JBossWS-CXFWSReliableMessagingTutorial>. Acesso em: 10 set. 2014.
11. Oracle. The Java EE Documentation. Disponível em:<http://www.oracle.com/technetwork/java/javaee/documentation/index.html>. Acesso em: 28 nov. 2017.
12. Angular Documentation. Disponível em:<https://angular.io/docs>. Acesso em 28 nov. 2017